06-02-18

Hibernate.

Q.*) Hibernate Configuration file. (setup file)

This is also called as setup file. It should be created with name "hibernate.cfg.xml" in Application.

It must contains 4 connection properties. Those are driverclass, url, ~~usr~~ username and password.

And it should also contain dialect (class also contain dialect (class generates SQL). show_sql and hbm2ddl.auto are optional.

eg:-

cfg file = property (key = value) + mapping class

Few keys details are :-

hibernate.connection.driver_class =
hibernate.connection.url =
hibernate.connection.username =
hibernat.connection.password =
hibernate.dialect =
hibernate.show_sql =
hibernate.hbm2ddl.auto =
(create | update | create-drop | validate)

** Here word "hibernate." is optional

* Cfg file format

```
<Hibernate-configuration>
<session-factory>
<property name="key">
        value </property>
<mappling class =
        "modalClassName"/>
```

Scanned by CamScanner

```
</session-factory>
</hibernate configuration>
```

* Cfg with Oracle DB Example:-

```
< hibernate configuration>
< session-factory>
< property name ="hibernate.connection.driver_class">
        oracle.jdbc.driver.OracleDriver </property>
< Property name="hibernate.connection.url"> jdbc:oracle:
        thin:@localhost:1521:xe </property>
<Property name="hibernate.connection.usename ~~password~~"> system
        </property>
<Property name="hebernate.connection.password">
        manager </property>
<property name="hibernate.dialect"> org.hibernate
        dialect.OracleDialect </property>
< property name="hibernate.show_sql"> true </property
< property name="hibernate.hbm2ddl.auto"> create
        </property>
< mapping class="com.app.Emp">
  </session-factory>
</hibernate-configuration>
```

Scanned by CamScanner

```
<hibernate configuration>
<session-factory>
<property name="hibernate.connection.~~mysqlDriver~~"
        driver_class> com.mysql.jdbc.Driver </property>
<property name="hibernate.connection.url"> jdbc:
        mysql://localhost:3306/test </property>
<property name="hibernate.connection.username">
        root </property>
<property name="hibernate.connection.password">
        root </property>
<property name="hibernate.dialect">org.hiberal
        dialect.MySQLDialect </property>
<property name="hibernate.show_sql"> true
        </property>
<property name="hibernate.hbm2ddl.auto">
        update ~~Create~~ </property>
<mapping class="com.app.emp">
</session-factory>
</~~conf~~ hibernate configuration>.
```

07-02-18

* Hibernate Test class :-

This class is used to perform DB operations like save, update, delete and select.

* Here operations are considered as 2 Types.
  1. select operation.
  2. non-select operations (insert, update, delete)

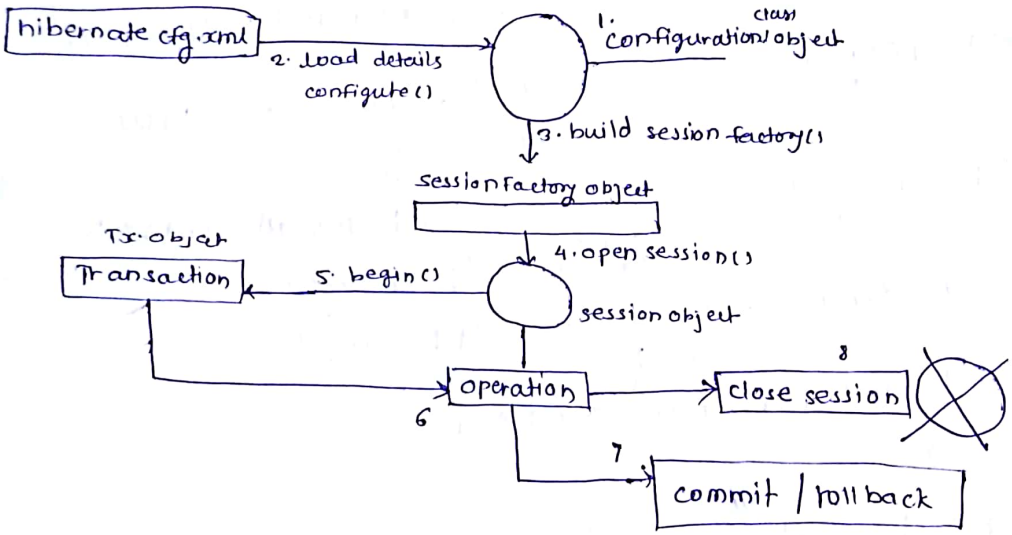
Scanned by CamScanner

* Transaction should be applied for non-select operations begin-tx then commit/rollback.

* For select operation Tx is not repli required. Even applied by programmer, gives no meaning (no error/no exception).

* Here sessionfactory (I) will perform
  → loading driverclass.
  → connection management
    (create, maintain, close connection object).
  → statement object management.

* Test class Flow :-



1) create configuration class object

Code:
```
configuration cfg = new configuration();
```

Load configuration XML file into cfg object using

```
cfg.configure();
```

3) Build one sessionFactory (I) object using cfg object.

```
SessionFactory sf = cfg.buildSessionFactory();
```

4) open session using sf object.

```
Session ses = sf.openSession();
```

5) ~~being~~ begin transaction using session.

```
Transaction tx = ses.beginTransaction();
```

6) Perform the operation using session object.
ex:, save(), update(), delete(), get(), load() .....

7) commit / Rollback Tx is created.

```
tx.commit() / tx.rollback();
```

8) close session at last.

```
ses.close();
```

* **Example ~~Code~~ codes with imports :-**

```
package com.app;
// ctrl+shift+o (imports)
import org.hibernate.*;
import org.hibernate.cfg.*;
public class Test {
    P S V main (String [] srs) {
        configuration cfg = new configuration ();
        cfg.configure ();
        SessionFactory sf = cfg.buildSessionFactory ();
        Session ses = sf.openSession ();
```

Scanned by CamScanner

```
Transaction tx = session.beginTransaction();
// operation..
tx.commit();
        or
// tx.rollback();
session.close();
}
}
```

08-01-18

* Download Hibernate Jars:-

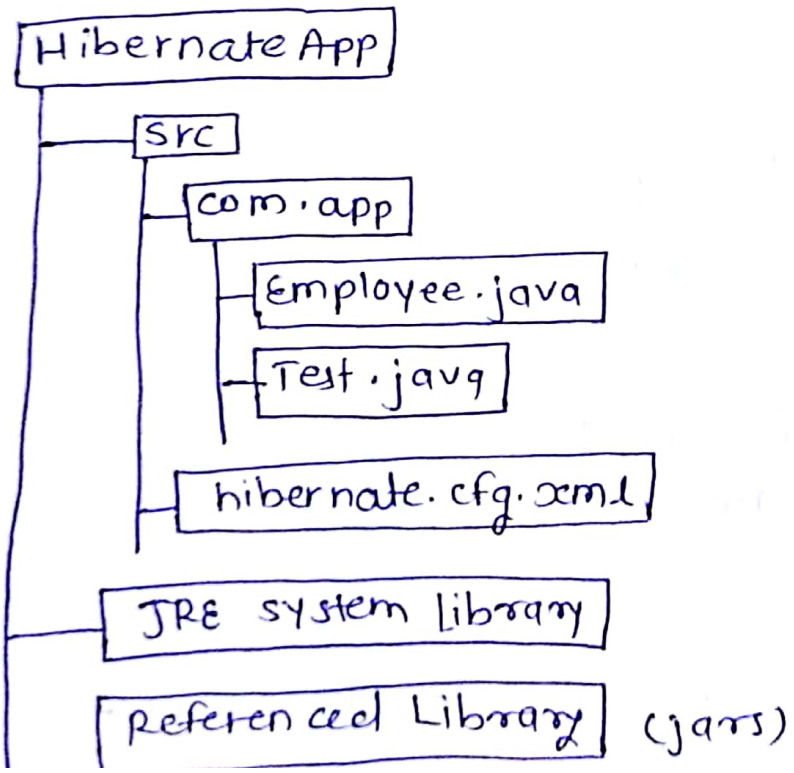
http://sourceforge.net/project/hibernate/files/hibernate3/3.6.10.final/

* Click of this option:

"hibernate-distribution-3.6.10.final-dist.zip".

* Right click and extract to one folder.

* Eclipse folder structure:-



Scanned by CamScanner

Prakash Reddy

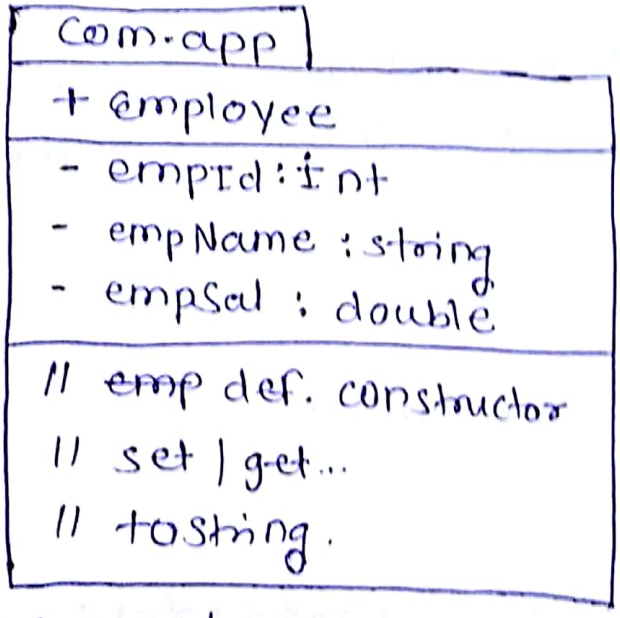
* Setup :-

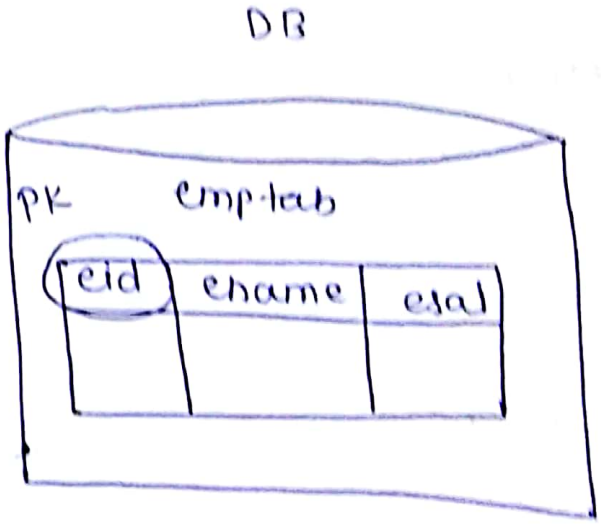
1) Open Eclipse and change perspective to "Java".
→ windows → open Perspective → other → choose "Java"

2) create "Java Project".
→ File → new → Java project → enter name "Hibernate".
→ finish.

3) Add jars to project.
→ Right click on project → build path → configure build path → click on "Library" Tab → click "Add External Jars" → select Hibernate + DB jars → finish.

* ~~Desi~~ Design :-





Protected
+ public
- private

Scanned by CamScanner

* Code:-

1) Model class :-

```
Package com.app;
// ctrl + shift + o (imports)
@ Entity
@ Table (name = "emptab")
public class Employee {
  @Id
  @ Column (name = "eid")
  private int empId;
  @column (name = "ename")
  private ~~int~~ string empName;
  @ column (name = "esal")
  private ~~int~~ double empsal;

// Alt + shift + s , O (Deselect All > ok) → Default constructor
// Alt + shift + s , R (select All > ok) → setter and get methods.
// Alt + shift + s , s (ok) → tostring.
}
```

2) Hibernate configuration File :- (xml file)

* hibernate.cfg.xml

→ right click on "src" → new → other → Type "XML FILE"
→ select "XML FILE" → ~~select~~ next → Enter name hibernate.cfg.xml. → finish.

Scanned by CamScanner

* code-(cmd file) :-

```
<!DOCTYPE hibernate-configuration PUBLIC.
"-//Hibernate/Hibernate configuration DTD 3.0//EN"
" http://www.hibernate.org/dtd/hibernate-configu
ration-3.0.dtd">
<hibernate-configuration>
<session-factory>
<property name="hibernate.connection.driver_cl
> oracle.jdbc.driver.OracleDriver </property
<property name="hibernate.connection.url">
jdbc:oracle:thin:@localhost:1521:xe </pro
<property name="hibernate.connection.username">
System </property>
<property name="hibernate.connection.passwor
manager </property>
<property name="hibernate.dialect"> org.hibern
dialect.OracleDialect </property>
<property name="hibernate.show-sql"> true
</property>
<property name="hibernate.hbm2ddl.auto"> cre
</property>
<mapping class="com.app.employee">
</session-factory>
</hibernate-configuration>
```

Scanned by CamScanner

3) Test class :-

```
package ~~ap~~.com.app;
// ctrl + shift + o (import)
Public class Test {
    Public static void main (String [] args) {
    Configuration cfg = new configuration ();
    cfg. configure ();
    sessionFactory sf = cfg. buildSessionFactory ();
    Session ses = sf. opensession ();
    Transaction tx = ses. beingTransaction ();
    Employee emp = new Employee ();
    emp. setEmpId (100);
    emp. setempName ("AB");
    emp. setEmpsal (12.36);
    ses. save (emp);
    tx. commit (); ses. close ();
    }
}
```

(ctrl + F 11)

→ Run Menu → run option

Scanned by CamScanner

09-02-18

* Database operations in hibernate using session:- (I):-

Session (I) supports two types of operations in hibernate. Those are select operations and non-select operations.

1) Non-select example operations are:

1) save (object) : serializable.
2) update (obj) : void
3) saveOrUpdate (obj) : void
4) delete (obj) : void.

* Select operations ~~on~~ are:

1) get (T.class, Id) : T Object.
   (T = model className)
2) load (T.class, Id) : T object.
   (T = model ClassName)

* 1) Save(obj) : Serializable :-

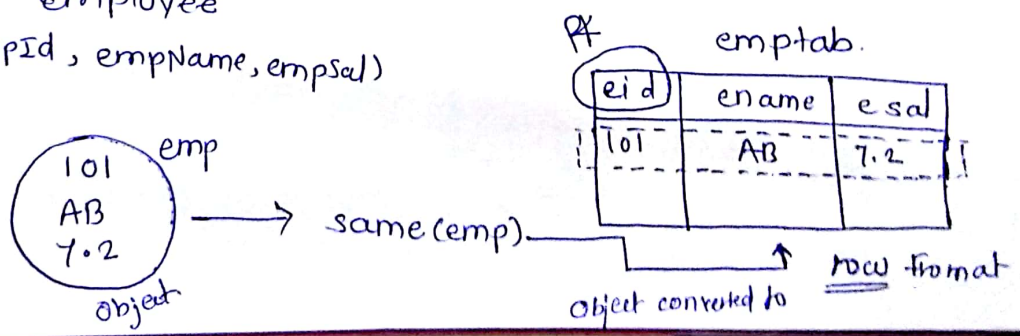
This method is used to convert object to row formed. This object should be model class (persistable class) object.

* when programmer write save() hibernate generates SQL look's as below:

insert into <table> values (....)

example:-

employee
(empId, empName, empSal)



Scanned by CamScanner

x) save(obj) method may throw exception if same primary key object (with different or same data) is given again.
i.e. constraintViolationException message is:
unique constraint violated.

Scanned by CamScanner

10-02-18

1) In model class annotations are from package. "Javax.persitance"

2) hibernate.cfg.xml should be created under "src" folder only.

3) <!DOCTYPE..> lines copy from source (Document given)

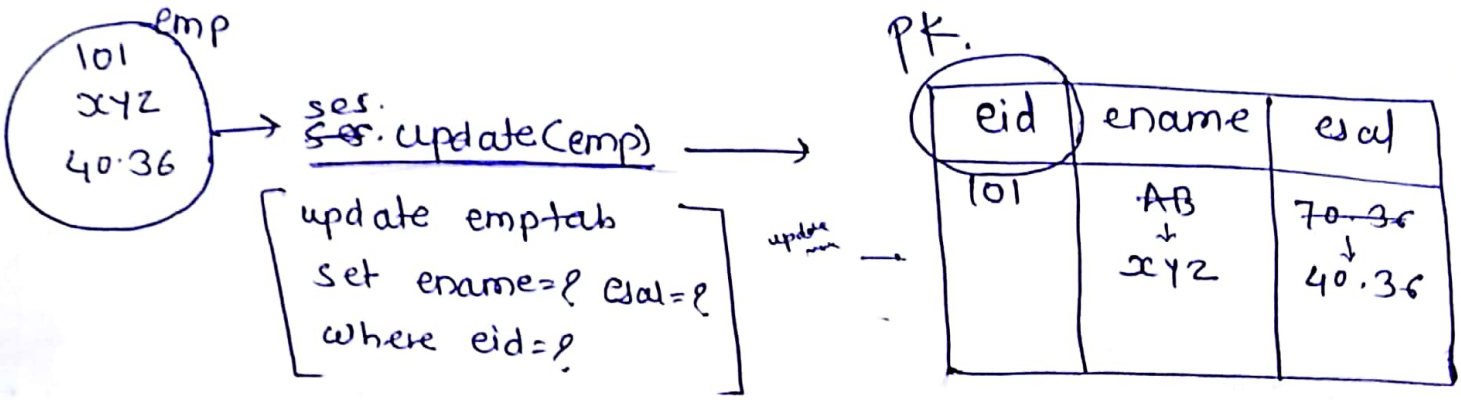
4) Setter / getter must be provided for all variables.

2) Update (obj) : void :-

This method is used to ~~all~~ update all columns value based on primary key column value.

*) If Record (Row) found based on primary key it updates else does nothing (no exception / no error).

example:-



* Code:-

1. Model class
2. hibernate.cfg.xms are same as before.
3. Test class:

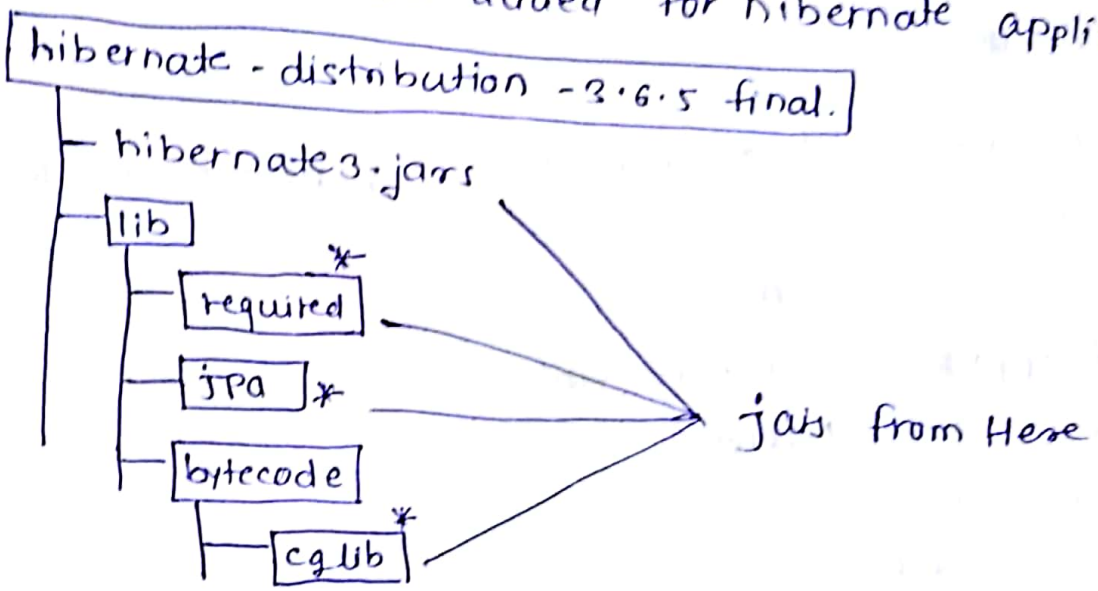
// cfg, sf, ses, Tx.

Scanned by CamScanner

```
Employee emp = new Employee();
emp.setEmpId(100);
emp.setEmpName("ABC");
emp.setEmpSal(12.36);
ses.update(emp);
tx.commit();   ses.close();
```

* Run Test class → ctrl + F11
(Run menu > Run option).

~~Either~~ Jar's to be added for hibernate application.



* Format_sql(boolean):-

It is a boolean property. default value is false. If it is true it will display SQL query in clause by clause (part by part).

To enable this add below key in hibernate.cfg.xml

```
< property name="hibernate.format_sql"> true
                                        </property>
```

* It works when show_sql is true.
Else this line has no meaning.

Scanned by CamScanner

example SQL :-

```
update
    emptab
set
    ename = ? , esal = ?
where
    eid = ?
```

* cfg.configure():-

This method will search for "hibernate.cfg.xml" in src only.

If location is different (.

(ex: com/appl hibernate.cfg.xml)

or name is different

(ex. sample.cfg.xml)

or both different.

(ex: com/app/sample.cfg.xml)

then use cfg.configure(string) ~~overloade~~

(overloaded) method.

* Example code :- (Test class).

```
configuration cfg = new configuration();
cfg.configure("com/appl hibernate.cfg.xml")
cfg.configure("sample cfg.xml");
cfg.configure("com/appl sample.cfg.xml");
sessionfactory sf = cfg.buildsessionfactory();
```

Scanned by CamScanner

12-02-18

* Temporary memory in Hibernate:-

Hibernate supports two types of temporary memories. Those are

i) cache ii) pool

Both are used to improve the performance of application.

1) Cache:-

It used in hibernate to reduce network calls between Hibernate App. and Database.

* Here, one-commit() = one_network call. ie

cache are two types in hibernate, these are,

1) session (I level) cache
2) session factory (II level) cache

13-02-18

*1) Session cache:-

15-02-18

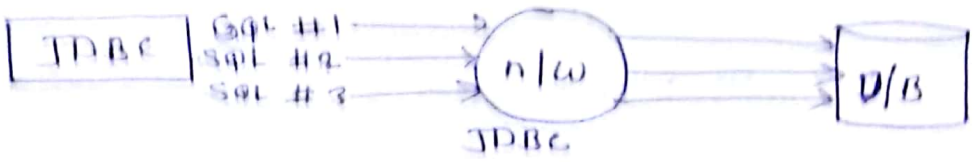
This memory is handled by hibernate only.

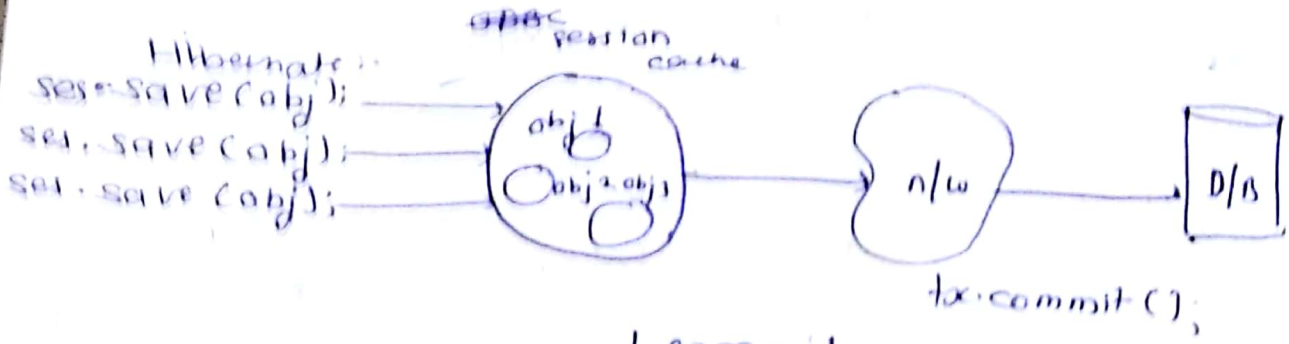
Programmer can not control this memory on performing any session operation that goes to session cache, only on commit data will be sent to data base.

* It executes all or none.
* By default every operation is batch in hibernate
* If one operation is faild in batch then all are cancelled.

Scanned by CamScanner

3 SQL's = 3 n/w calls





1 commit = 1 n/w call

* Example code :-

i) model class :- ii) configure file

same as above

*2) Test class :- // cfg, sf, ses, tx

```
Employee emp1 = new employee ();
    emp1.setEmpId (503);
    emp 1.setEmpName ("A");
    Emp1.setEmpSal (9.9);

Employee emp2 = new Employee ();
    emp2.setEmpId (504);
    emp2.setEmpName ("B");
    emp2.setEmpSal (8.8);

Employee emp3 = new Employee ();
    emp3.setEmpId (505)
    emp3.setEmpName ("C");
```

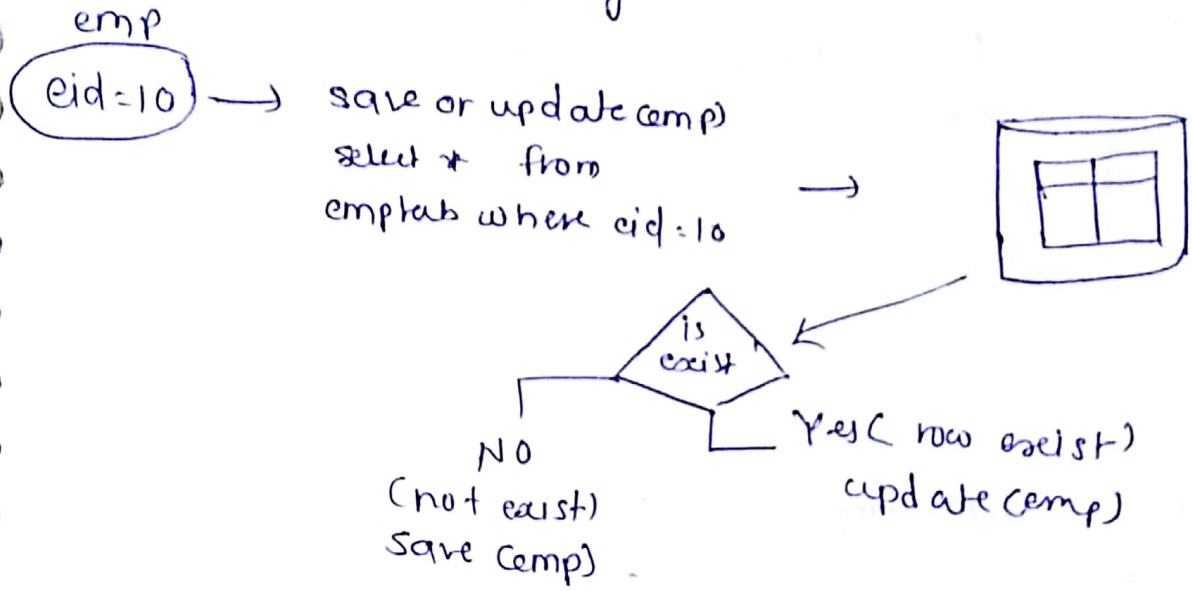
Scanned by CamScanner

```
emp3.setEmpSal(7.7);

ses.save(emp1);
ses.save(emp2);
ses.save(emp3);
tx.commit();
ses.close();
```

* save(); update(obj) :-

This operation executes select query first based on primary key, if row exist then it executes update query else insert query.



* Example code :-

i) Model class ii) hibernate.cfg.xml

Same as before

iii) Test class :- // cfg, sf, ses, tx

```
Employee e = new Employee();
    e.setEmpId(10);
    e.setEmpName("A");
    e.setEmpSal(6.6);
```

Scanned by CamScanner

ses.save or update(e);
tx.commit();

* ses.delete(obj);

This method is used to delete one row based on primary key. But input is "model class object having only primary variable value".

eg code:-

model class & cfg file same as above.

Test class :- //cfg, sf, ses, tx.

```
Employee e = new Employee();
e.setEmpId(50);
ses.delete(e);
tx.commit();
ses.close();
```

Employee emp

empId = 36 → ses.delete(emp) →

SQL:
delete from emptab
where eid = 36;

emptab

eid	ename	esal
36	A	50

14-02-18

Data select using get();-

session supports selecting one row based on Primary key column get() will select one row and that is converted to one object.

row > get() > object.

Scanned by CamScanner

* If row is not exist for given primary key then get() returns null data.

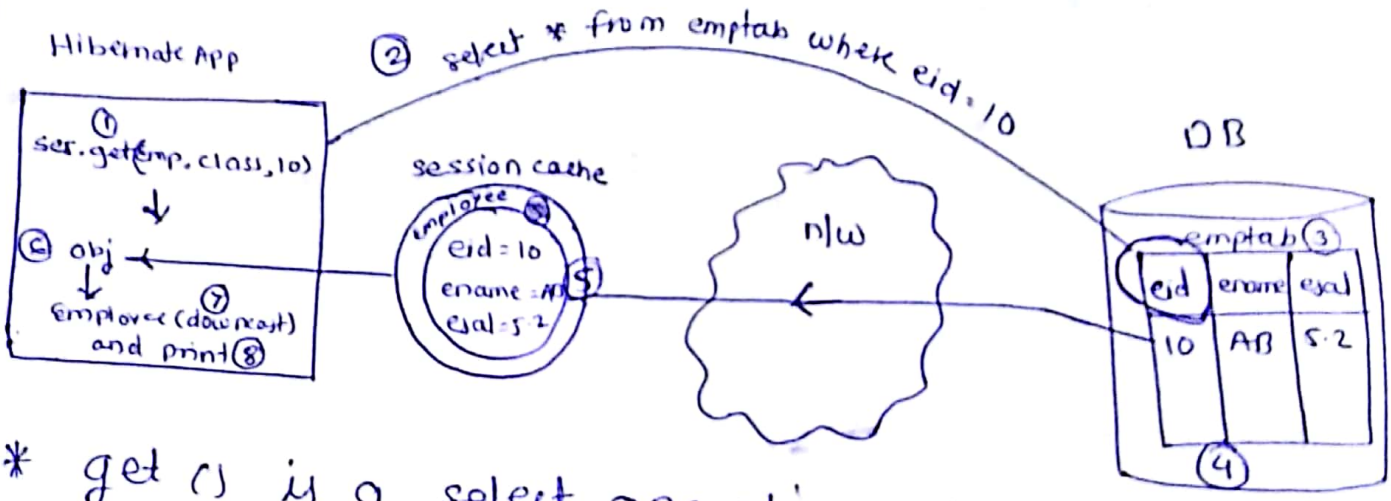
* get() works only based on primary key column, not other.

Syntax:-

get (T.class, ID) : object

Here, T = modelClass Name = Type
ID = PrimaryKeyValue.
Object = Row is converted to java.lang.object.

* get() flows:-



* get() is a select operation. so Tx is no required while writing this operation.

* get() always connect to DB to fetch data every time. It never goes to cache directly.

* get() returns null if row not exist.

* Example:-

① model class and hibernate.cfg.xml is same as above/before.

Scanned by CamScanner

* 2) Test class:- cfg/sf, ses

```
Object ob = ses.get(Employee.class, 10);
Employee e = (Employee)ob;
sout(e); ses.close();
```

* Q.) casting null data will throw any exception or not?

Ans: → If reference has null performing operation like method call, property call returns NullPointerException (NPE)

ex:-

```
Object ob = null;
ob.toString();   // → NPE
```

But casting (Down Cast) never return's exception, null casted value is null only

ex:-

```
Object ob = null;
Employee e = (Employee)ob;
sout(e);
```

o/p → null

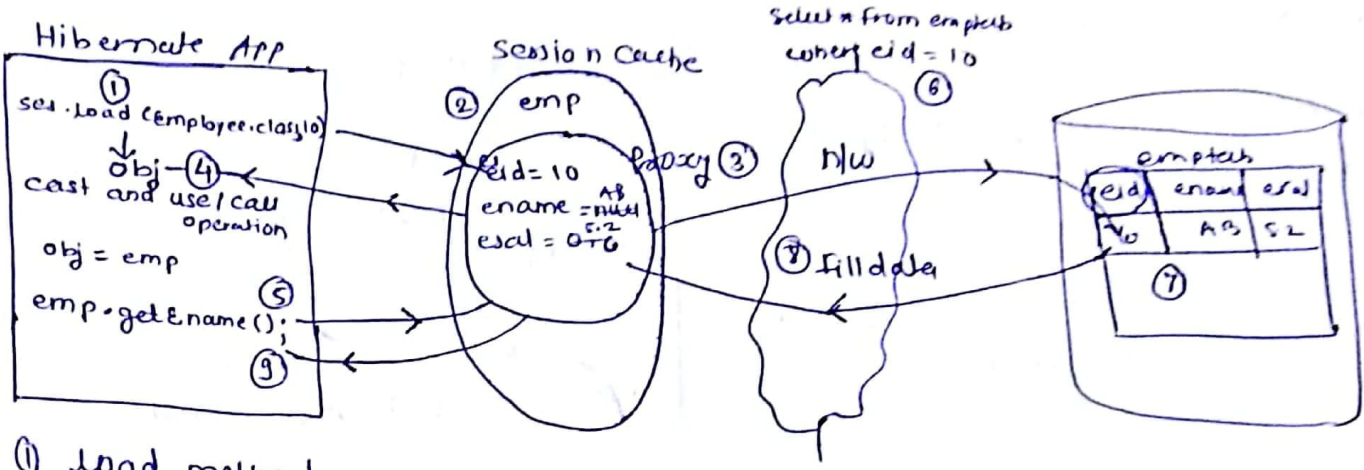
Load(T.class, ID): object

This method creates one proxy (dummy / fake object) in session cache and returns same to application. On performing any operation using object after asting then it goes to DB and fetch the data, files object variables.

Scanned by CamScanner

* If row is not available in DB then it returns exception

"Object Not Found Exception".



1. load method
2. session cache
3. proxy
4. return to hibernate object
5. operation.
6. make d/B
7. search row
8. Fill data in proxy
9. return to application

* Code :-

*1) model and cfg files same as before.

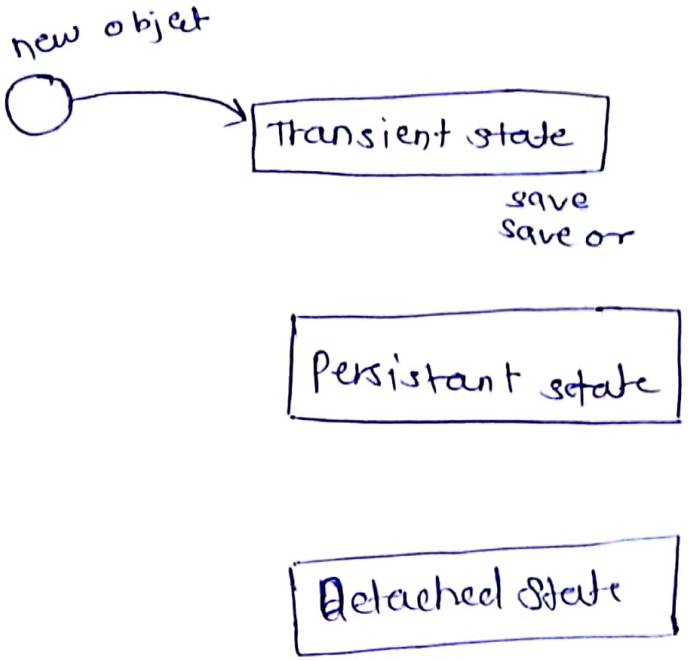
(*2) Test class : // cfg, sf, ses

```
object ob = ses.load(Employee.class, 51);
Employee e = (Employee) ob;
sout(e);
ses.close();
```

Scanned by CamScanner

Q) What is the difference b/w get() and load() method?

get()	load()
1) get() hits DB always	1) load() hits cache always.
2) get() return null if record not exist.	2) load throws exception if record not exist (ObjectNotFound)
3) every time it makes network call	3) only on reading data it makes network call..



) Transient state :-

An object exist in heap memory and not connected to any cache, then object set is nsient.

Scanned by CamScanner

ex:-

Employee e = new Employee ();

→ Object create in heap (JVM / RAM)
But not in cache.

*2) Persistence state:-

On performing session to operations like save, update, saveorupdate, get or load then object will be placed (connected) to cache, then object is in "Persistence state."

ex:-

Emp e = ~~emp~~ ses.get(Emp.class, 55);

*3) Ditached state:-

An object removed from cache which is eligible for garbage collection is in "Ditached state".

ex:-

ses.delete(emp);

* GC :- follows mark and sweep Algorithm.

* Annotations concept #3

* 3) DATE & TIME :-

In model class ~~java.~~ for date (java.util.Date) we can specify DB storage format using

Annotation	+	Enum
@Temporal		Temporal Type {DATE, TIME, TIMESTAMP}

Scanned by CamScanner

DATE : stores only Date like 11/02/2018
TIME : stores only Time like 09:35:25
TIMESTAMP : stores Date and Time like
11/02/2018 09:35:25

* Code :-

```
1# Model class :
Package com.app;
// ctrl + shift + o (imports)
@ Entity
@ Table (name = "emptab")
public class Employee {
  @Id
  @column (name = ~~eid~~ "emptab")
  private int empId ;

  @ Temporal (TemporalType.TIMESTAMP)
  @ column (name = "dte_a");
    private ~~String~~ Date dteA;

  @Temporal (TemporalType.TIME)
  @ column ( name = "dte_b").
    private Date dteB;

  @ Temporal (TemporalType.DATE)
  @ column( name = "dte_c")
    private Date dtec;
// def. constr, set/get , toString.
}
```

Scanned by CamScanner

* for mySQL DB cfg details are :

driver = com.mysql://localhost:3306/test

username = root

Password = root

dialect = com.hibernate.mySQLDialect

* Test class :// cfg, sf, ses, tx.

```
Employee emp = new Employee();
emp.setEmpId(1000);
emp.setDateA(new Date()); // java.util.
emp.setDateB(new Date());
emp.setDateC(new Date());
ses.save(emp);
tx.commit();
// load and print to see output.
employee e = (employee) ses.get(employee.class
                                , 1000);
Sout(e);
ses.close();
}
```

Scanned by CamScanner

16-02-18

* Secondary Table in Hibernate:-

By default ~~one~~ one class can be connected to one table ~~by~~ only. But hibernate supports few variables in class can be stored with 2nd table also called as secondary table. Specialy fields like. sum, total, avg final grades, export data logical values stored in another table. instead using main table for these conceptswe use secondary table.

* @Entity and @Table provides mapping to main table also called as primary Table.

*) @secondaryTable is used to provide mapping to 2nd, 3rd, 4th, 5th table also called as secondary table.

page no-9.

MATERIAL

```
Pjc  @secondary table (name = "emp_child",
     PkJoinColumn = @primaryKeyJoinColumn(
     name = "eidFk", referenceColumnName = "eid")
     @column(name = "eid")
     int empId;
     @column (name = "ename", table = "emp-child")
     String empName
```

required

optional

required

Scanned by CamScanner

* Example code :-

① Model class :-

```
Package com.app;
@Entity
@Table (name = "emptab")
@SecondaryTables ({
    @SecondaryTable (name = "emp info"),
    @SecondaryTable (name = "empdata")
})
public class Employee {
    @Id
    @Column (name = "eid")
    private int empId;

    @Column (name = "ename")
    private String empName;

    @Column (name = "esal", table = "empdata")
    private double empsal;

    @Column (name = "edsg", table = "emp info")
    private string empDesg;

    @Column (name = "edob")
    @Temporal (TemporalType.DATE)
    private Date empDate;

    @Column (name = "ebns", table = "emp info")
    private double eBonus;
```

Scanned by CamScanner

// def constructor, setlget, to string.

2) Hibernate.cfg.xml same as before.

3) Test class : // cfg, st, ses, tx.

```
Employee emp = new Employee();
emp.setEmpId(10);
emp.setEmpName("A");
emp.setEmpSal(1.6);
emp.setEmpDesg("MG");
emp.setEmpDob(new Date());
emp.empBonus(12.36);
```

tables:

emptab

eid	ename	edob
10	A	Feb 12, 2018 16-2-18

empinfo

eid	edgs	ebns
10	MG	12.36

empdata

eid	esal
10	1.6

* every table in hibernate must have primary key column. even secondaryTable also.

type-3 Component mapping

@Embeddable ← select the class

@Embedded ← This is connecting with Employee (required.)

@AttributeOverrides{
@—
@—
}

← Provides new columns name (optional)

MATERIAL

Scanned by CamScanner

* Component Mapping :-

Mapping multiple classes to one table if chil class is fully dependent on parent class.

Then use two annotations for selection @ e dable and for connection @ embeded.

Code:-

① Model class:-

```
Package com.app;
 // ctrl+shift+o (imports)
@Embeddable
 public class Address{
    @Column (name="aid")
    Private int addrId;
    @Column (name="loc")
    Private String loc;
// def const, set/get, toString }
```

2) Package com.app;

```
 // ctrl +shift+o (imports)
@Entity
@Table (name="emptab").
 public class Employee{
    @Id
    @Column (name="eid")
    Private int empId;
```

Scanned by CamScanner

```
@Column(name="ename")
private String empName;
@Column(name="esal")
private String empSal;

@Embedded
private Address addr; //Has-A relation
// default const, set/get, toString }
```

*(3) hibernate.cfg.xml, same as above

*(4) Test class :- // cfg, sf, ses, tx

```
Address a = new Address();
a.setAddrId(15);
a.setLoc("HYD");
Employee e = new Employee();
e.setEmpId(10);
e.setEmpName("AA");
e.setEmpSal(1.3);
~~e.setEmpN~~
e.setAddr(a);
ses.save(e);
tx.commit();
ses.close;
```

emptab				
eid	ename	esal	aid	loc
10	AA	1.3	15	HYD

Scanned by CamScanner